
StatsHouse:
метрики ВКонтакте
Григорий Петросян
HL
HighLoad++
2022

Hello, World

Я — Гриша, Core Infrastructure @ ВКонтакте

StatsHouse — наша система мониторинга

План

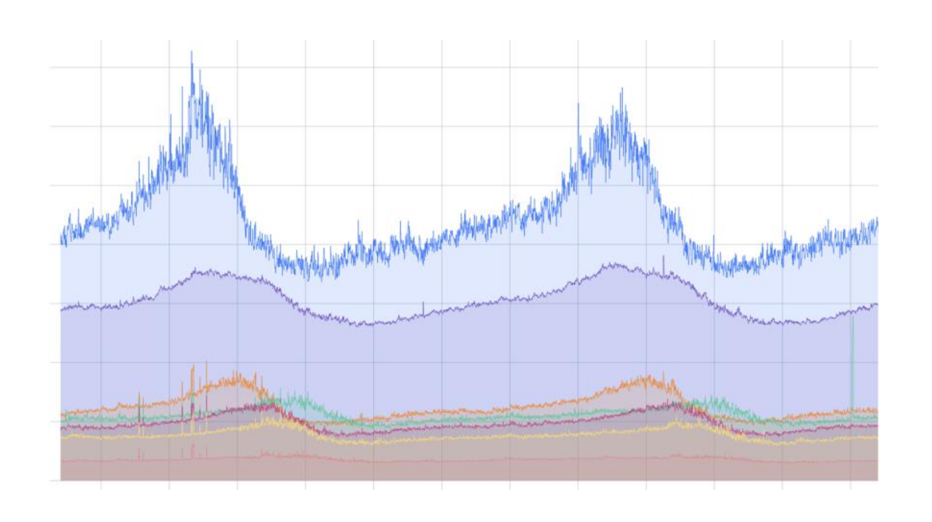
1. Про мониторинг *at scale*
2. Как эффективно собирать и хранить метрики
3. Как не дать сломать мониторинг
4. Как устроен StatsHouse
5. Анонс!

Мониторинг *at scale*



Мониторинг

```
cdn_ttfb {
    country = ru
     region = msk
         os = android
} = 0.239
```



Без мониторинга



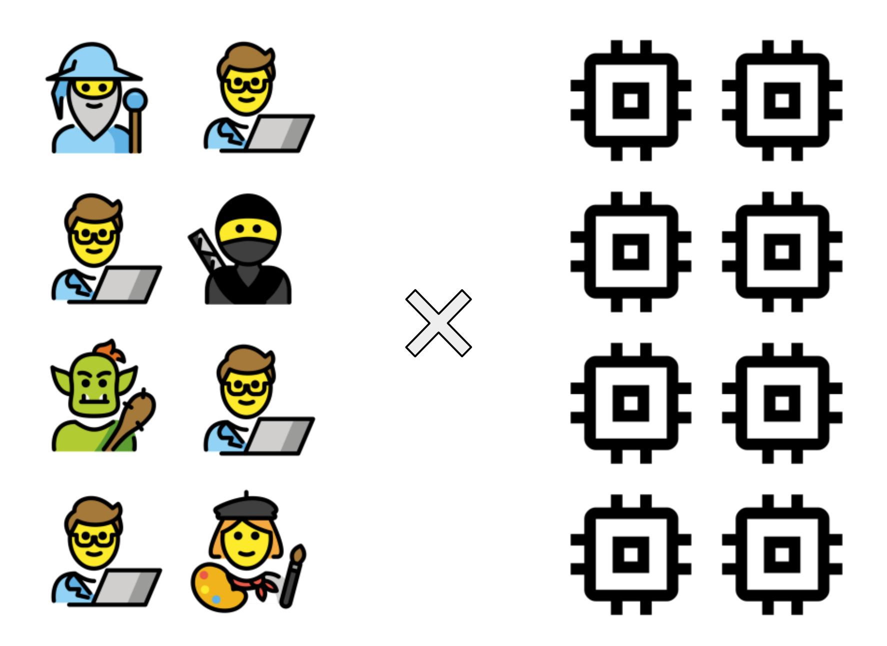
Мониторинг

Обязан работать всегда

Обязан работать быстро

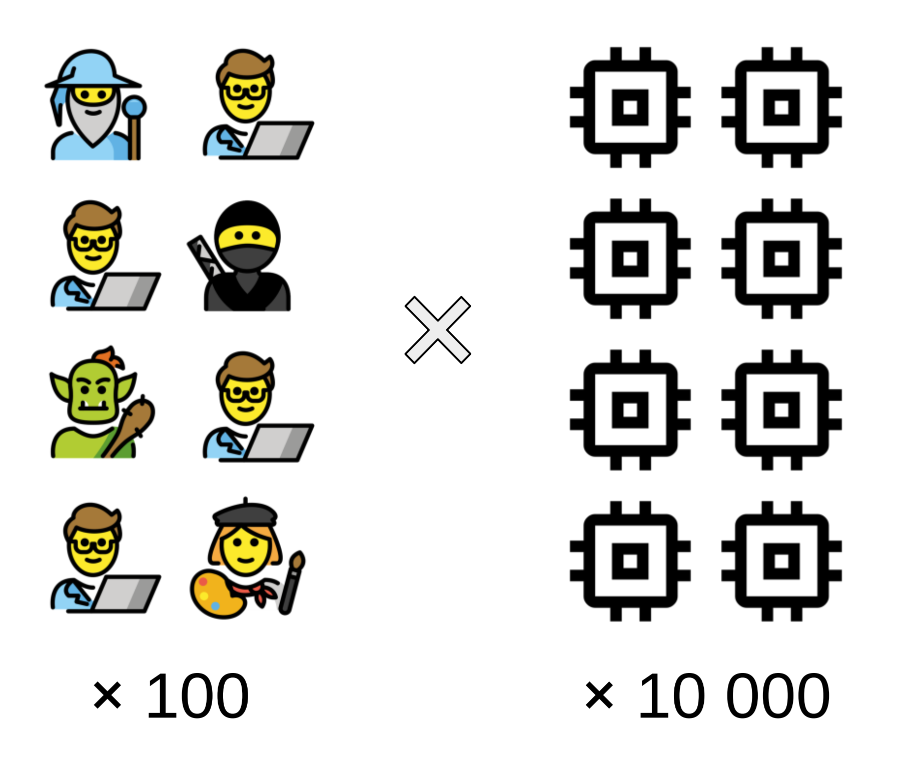
- Низкая задержка
- Высокое разрешение
- Интерактивность

Мониторинг? Prometheus





Prometheus не масштабируется





350 000 000

метрик в секунду с 15 000 серверов собирает StatsHouse


SONY
E-mount
α
7R

SONY
E-mount
α
7R

SONY
E-mount
α
7R

SONY
E-mount
α
7R

SONY
E-mount
α
7R

SONY
E-mount
α
7R

Мониторинг *at scale*



Унести все, за 1 секунду — каждую секунду

Быстрый путь
~~в яхт-клуб~~ домой



Метрики прекрасно сжимаются

```
cdn_ttfb {
    region = msk/spb/nsk
} ← (count: 1, value: dt)
```

6M RPS

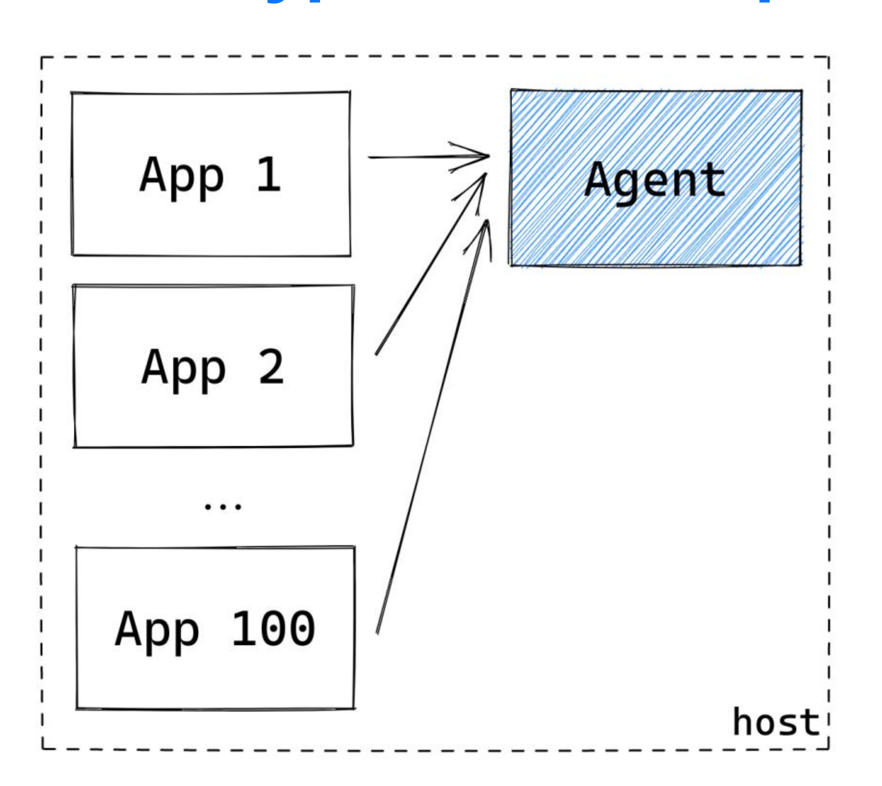
6M записей в секунду

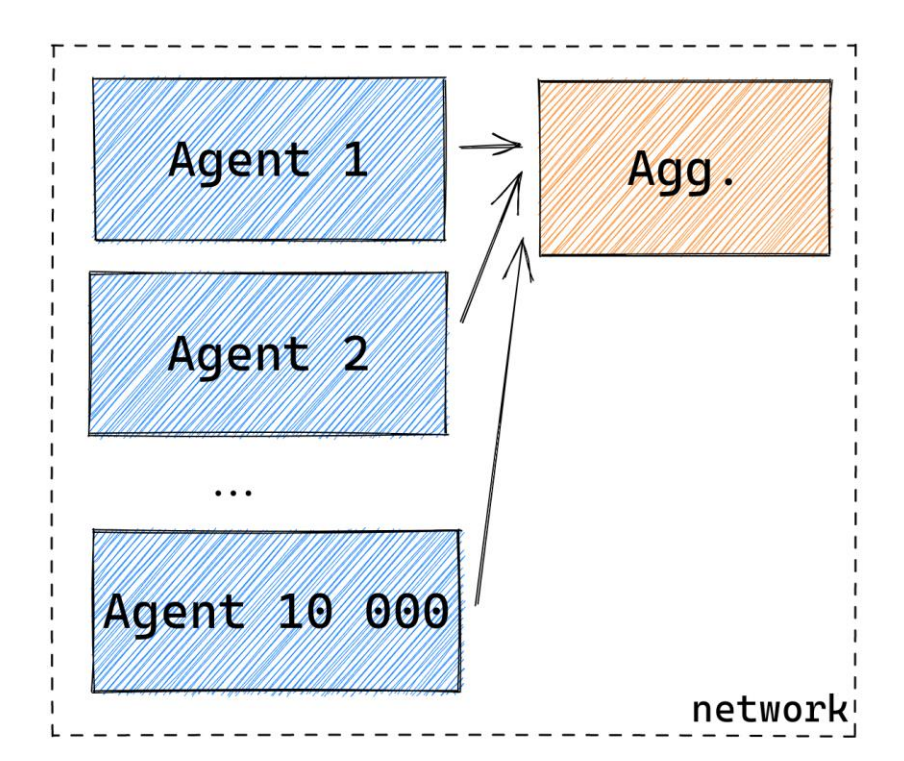
```
cdn_ttfb{msk} ← (3M, ∑dt)
cdn_ttfb{spb} ← (2M, ∑dt)
cdn_ttfb{nsk} ← (1M, ∑dt)
```

6M RPS

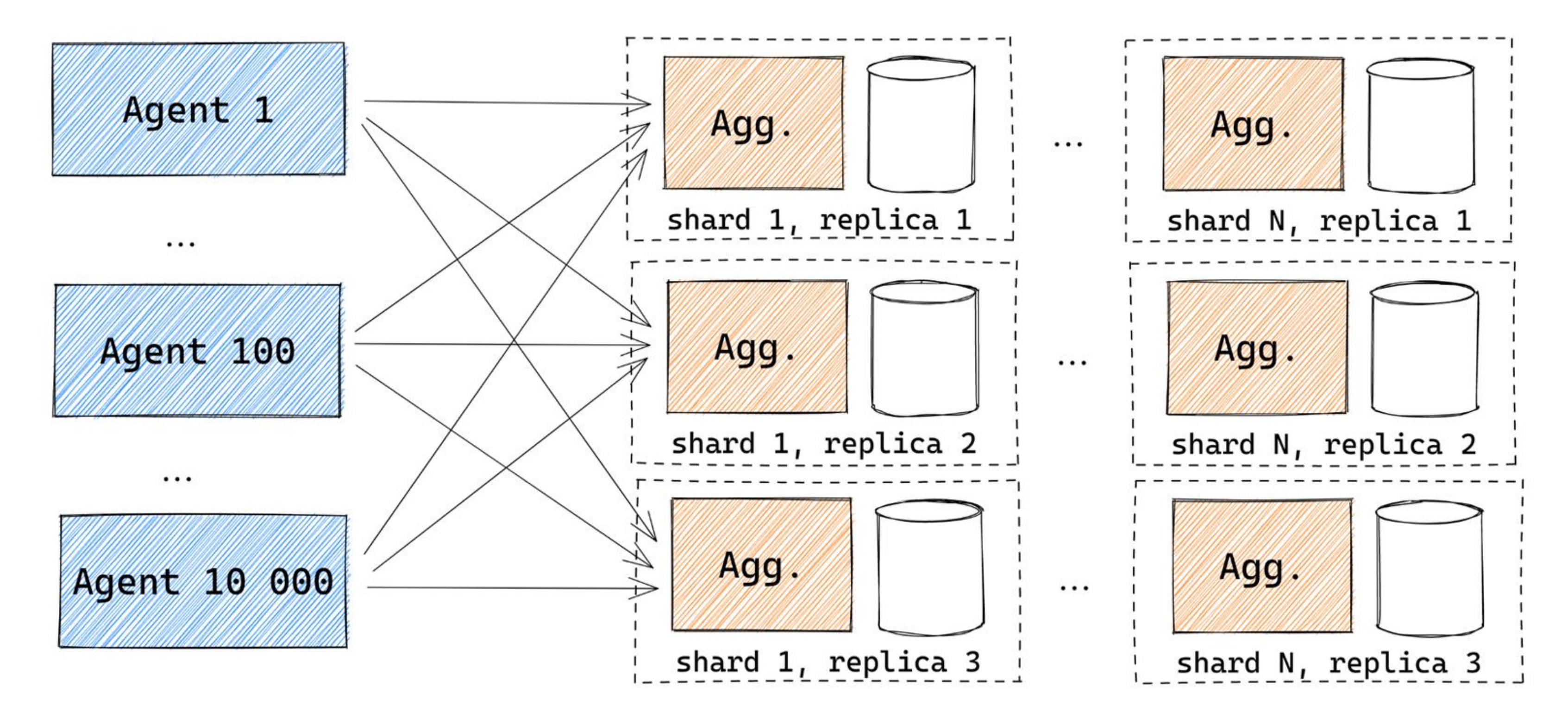
3 записи в секунду

Многоуровневая агрегация

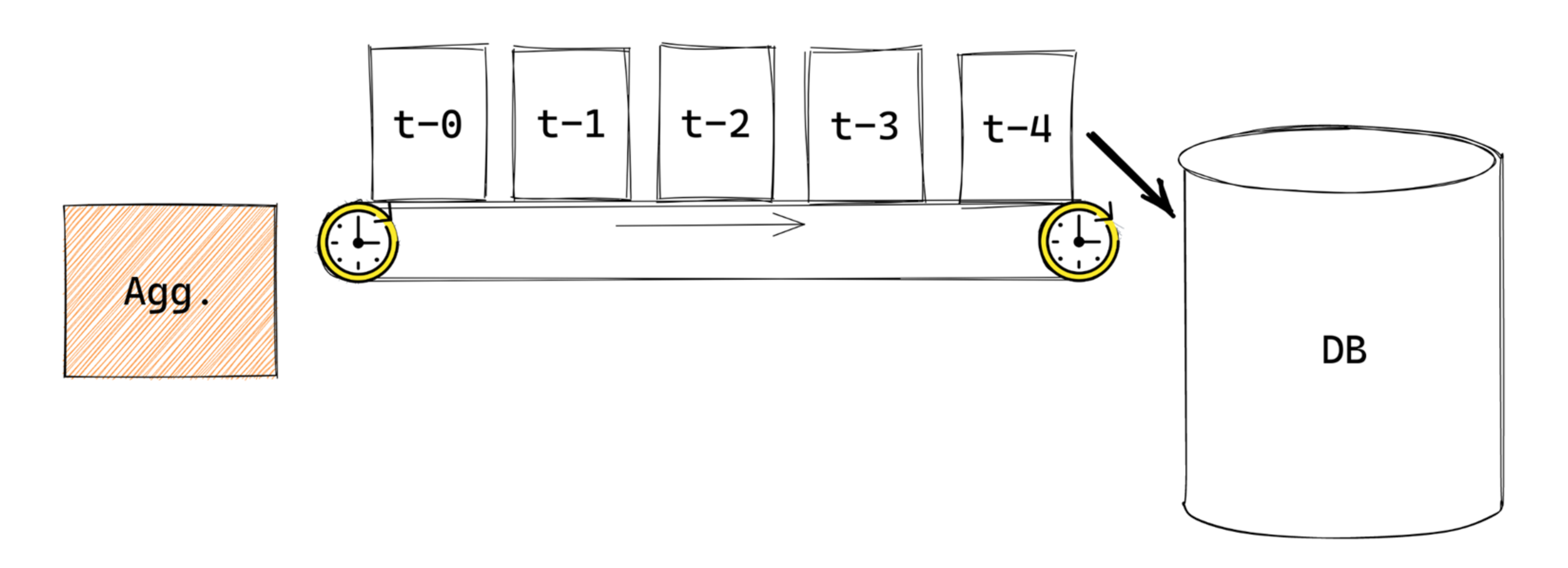




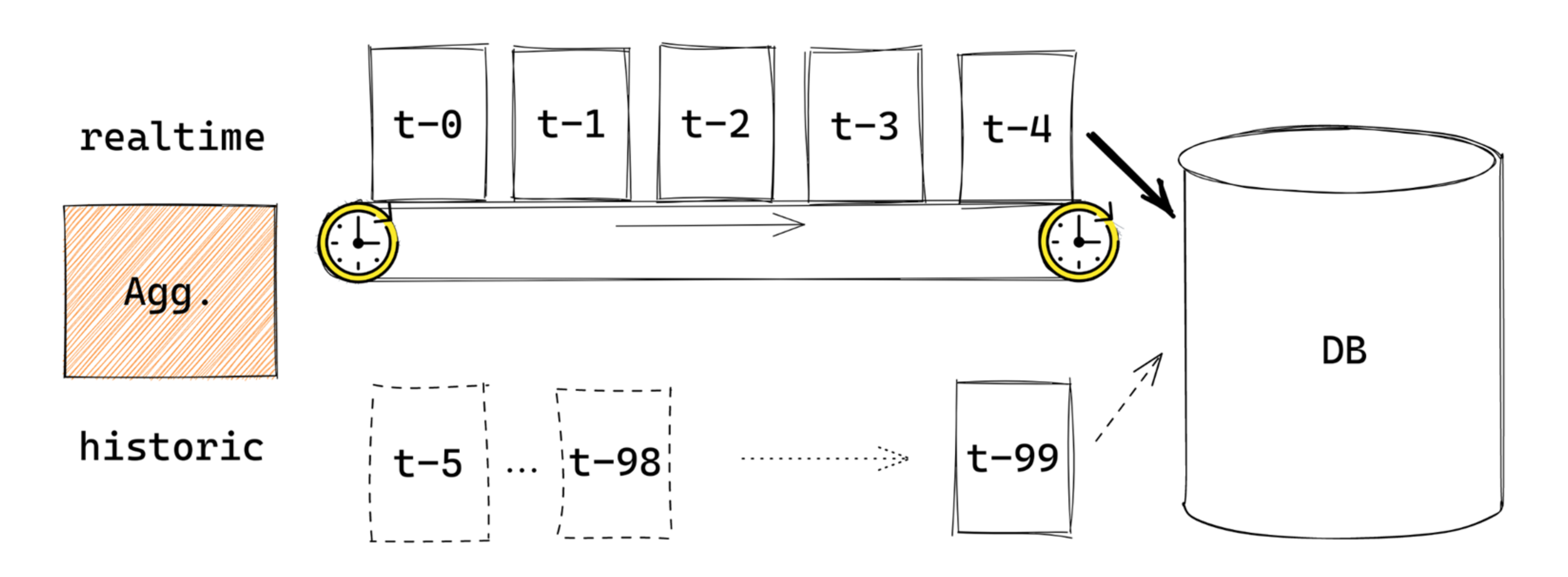
Шардирование и репликация



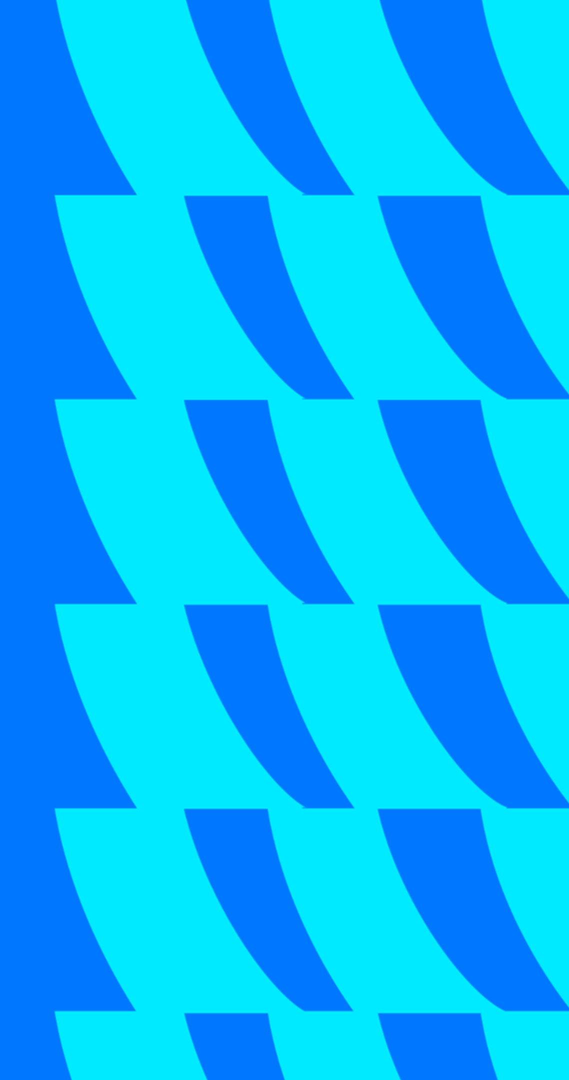
Конвейер записи



Конвейер записи: исторические данные



Не дать сломать мониторинг



Метрики прекрасно сжимаются

```
cdn_ttfb {
    region = msk/spb/nsk
} ← (count: 1, value: dt)
```

6M RPS

6M записей в секунду

```
cdn_ttfb{msk} ← (3M, ∑dt)
cdn_ttfb{spb} ← (2M, ∑dt)
cdn_ttfb{nsk} ← (1M, ∑dt)
```

6M RPS

3 записи в секунду

Метрики прекрасно разжимаются

`foo{…}`

100 Мбайт/с



`foo{…, $server}`

100 Гбайт/с



Метрики ~~прекрасно~~ бесконечно разжимаются

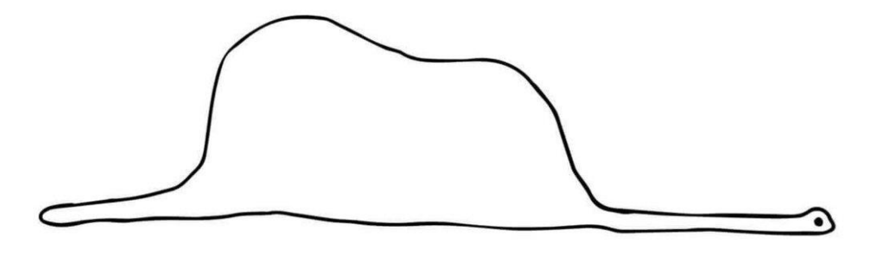
```
foo{…, $user_id, $url}
```

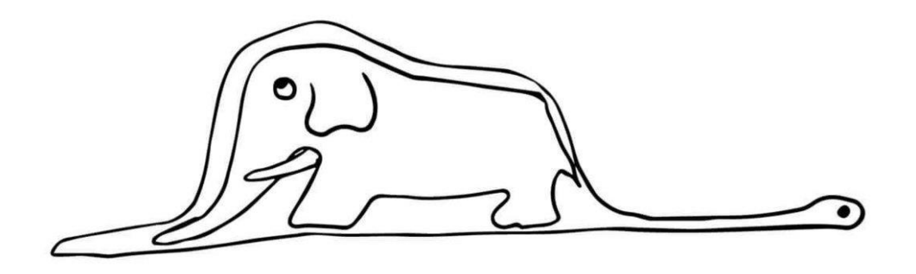
100 Тбайт/с



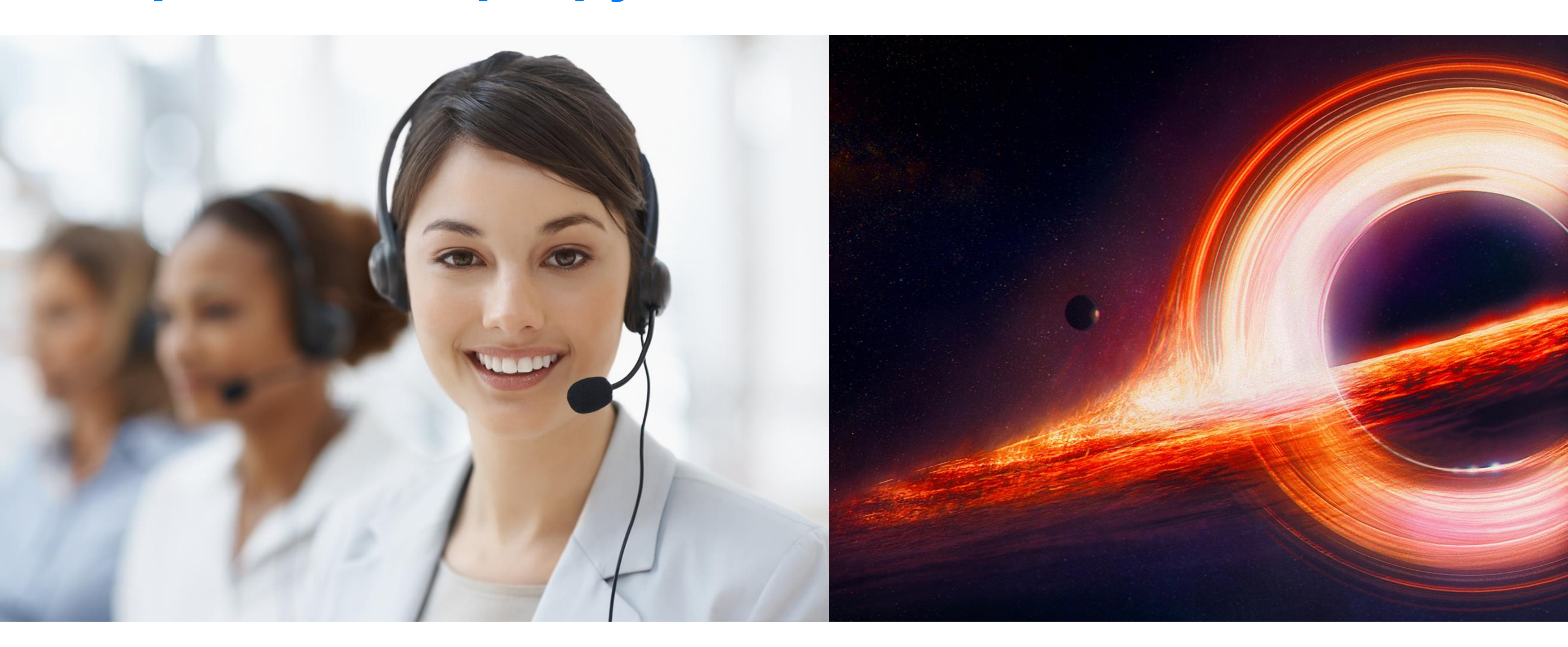
Kafka не спасет



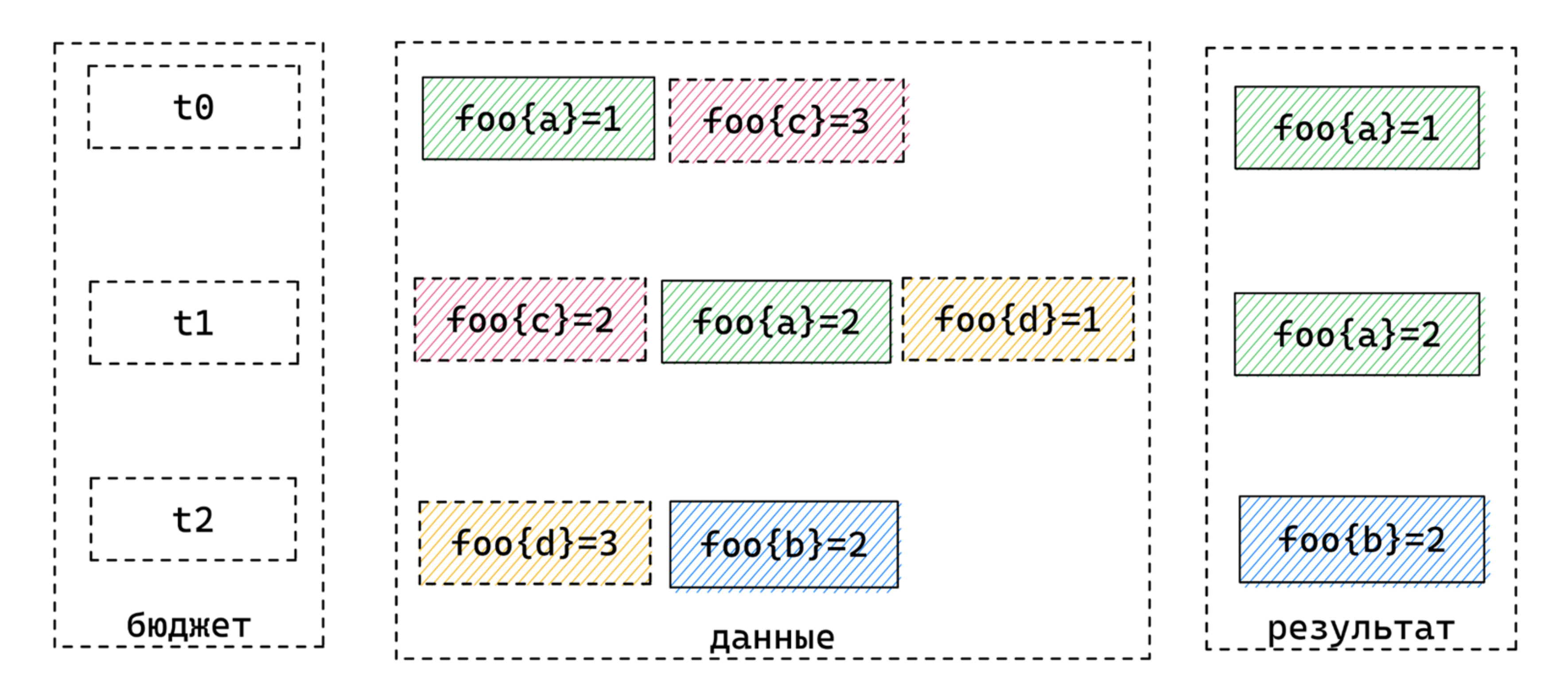




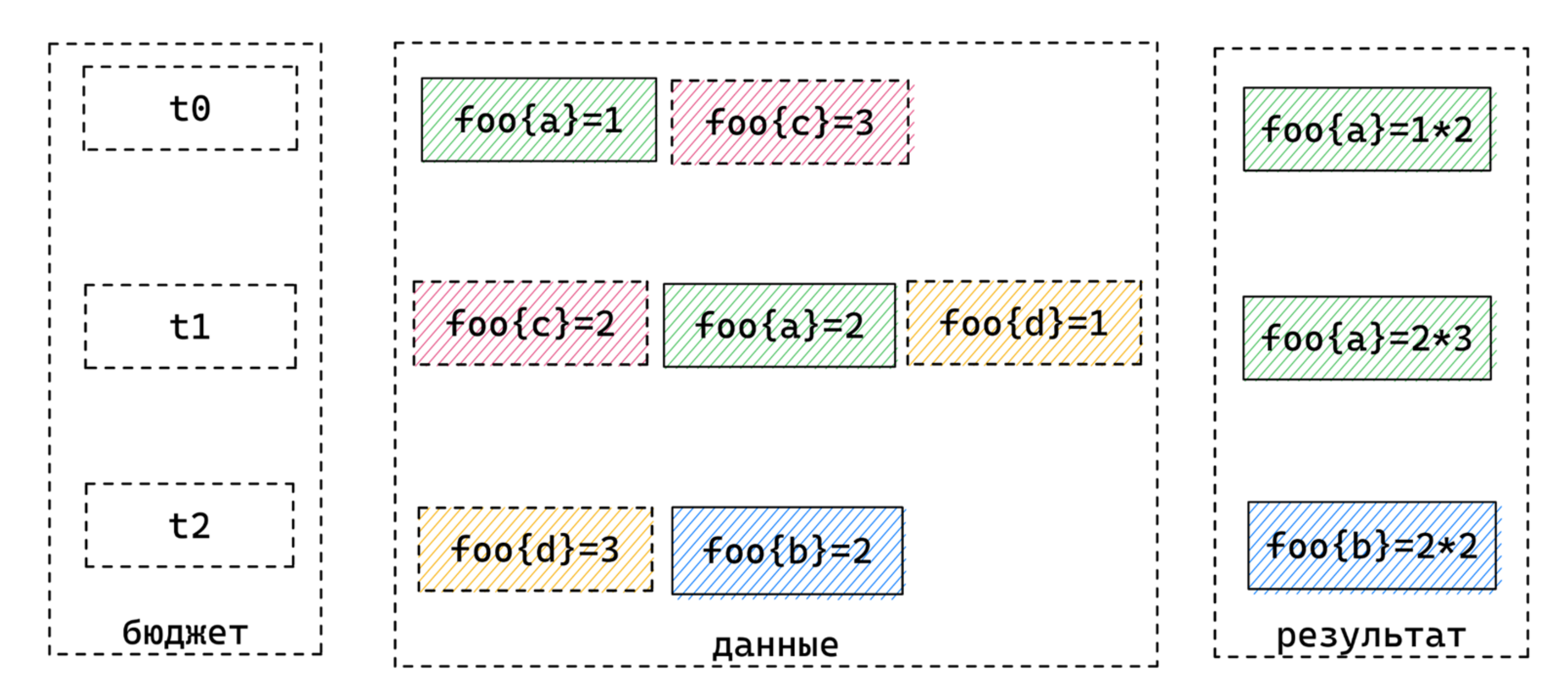
Боремся с перегрузкой



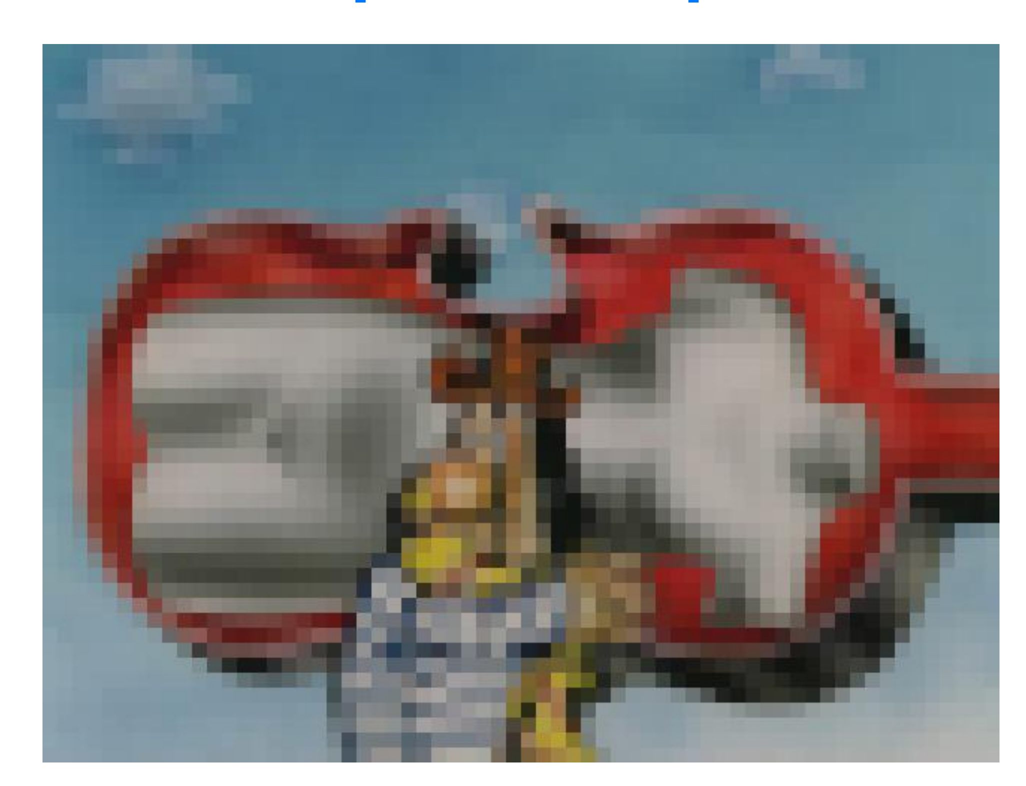
cat data > /dev/null



Семплирование



Семплирование работает



Не дать сломать ~~мониторинг~~ ничего

1. Бюджетируем ресурсы
   - Сеть
   - Диск
   - CPU
2. Честно распределяем между пользователями
3. Авто-семплируем при превышении бюджета

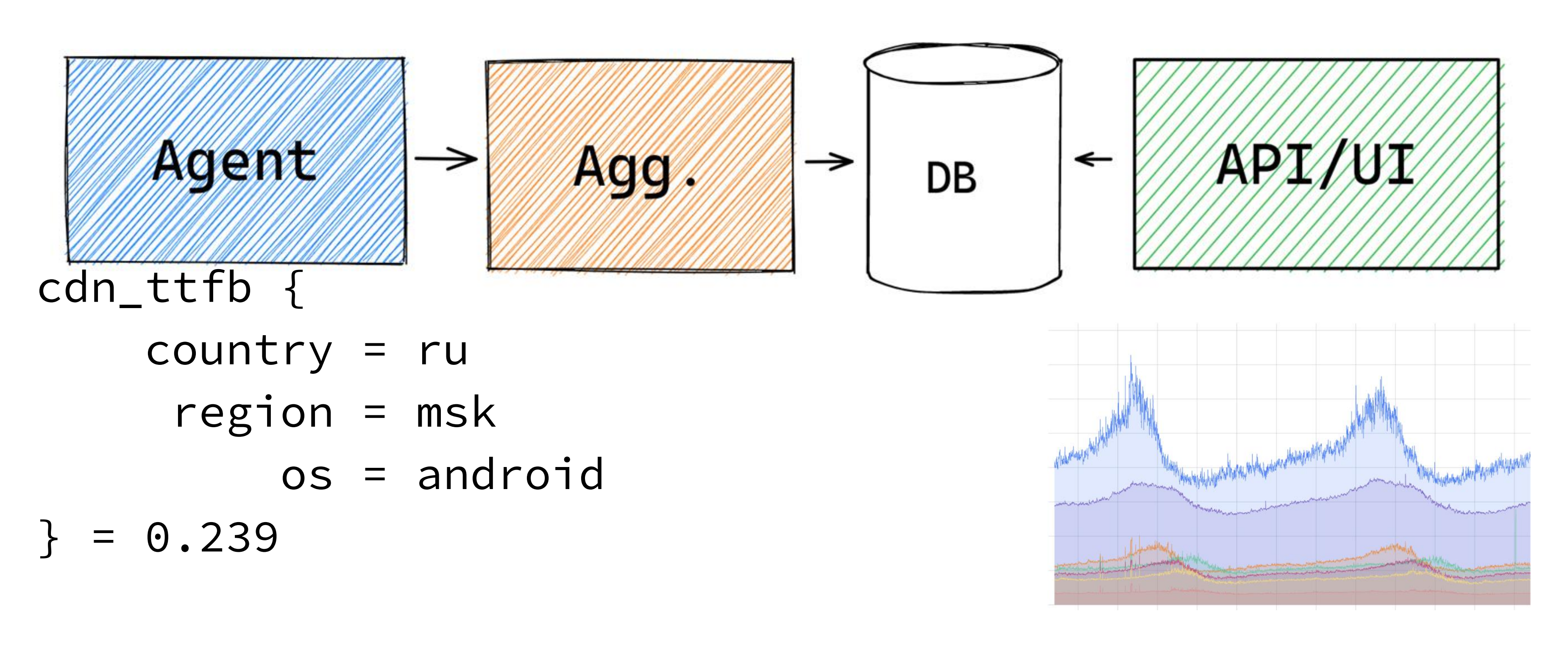
“Look Ma, a Highly-Available Multi-Tenant System!”



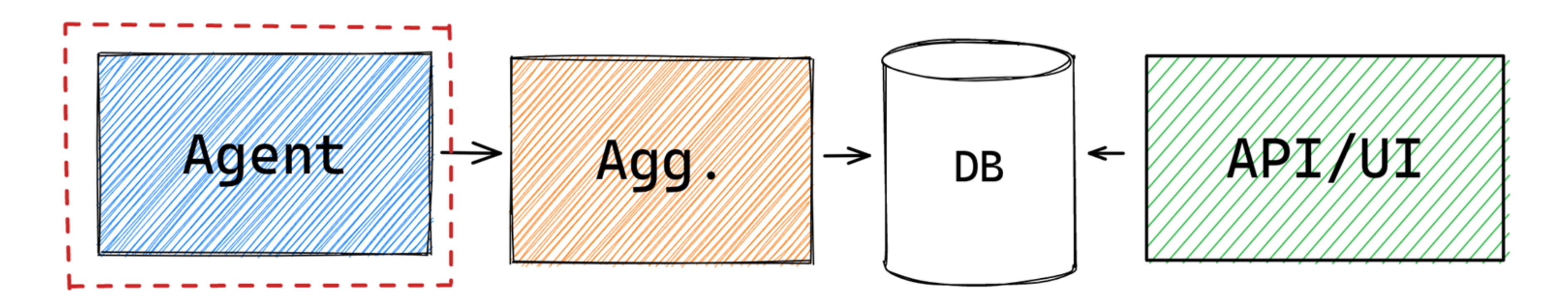
Как устроен StatsHouse



StatsHouse

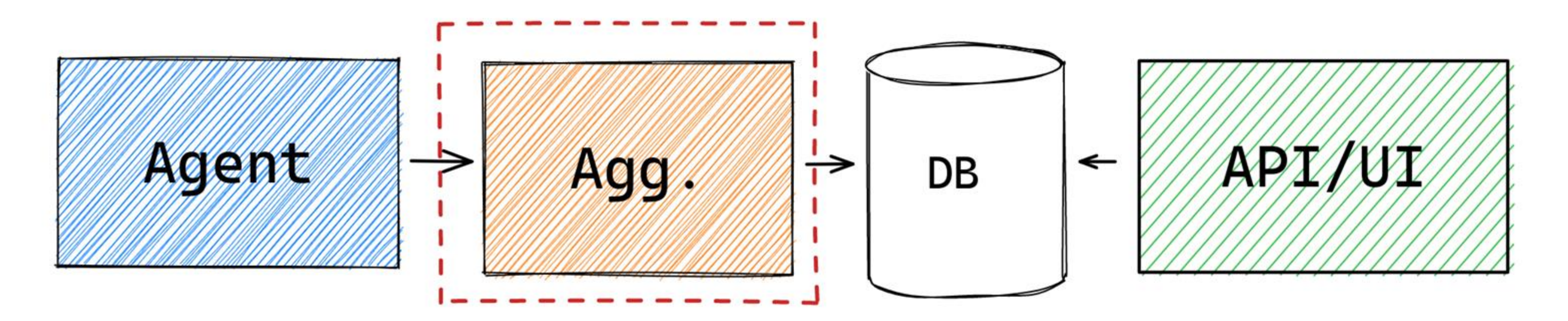

Agent
Agg.
DB
API/UI
cdn_ttfb {
country = ru
region = msk
os = android
} = 0.239

StatsHouse: агент



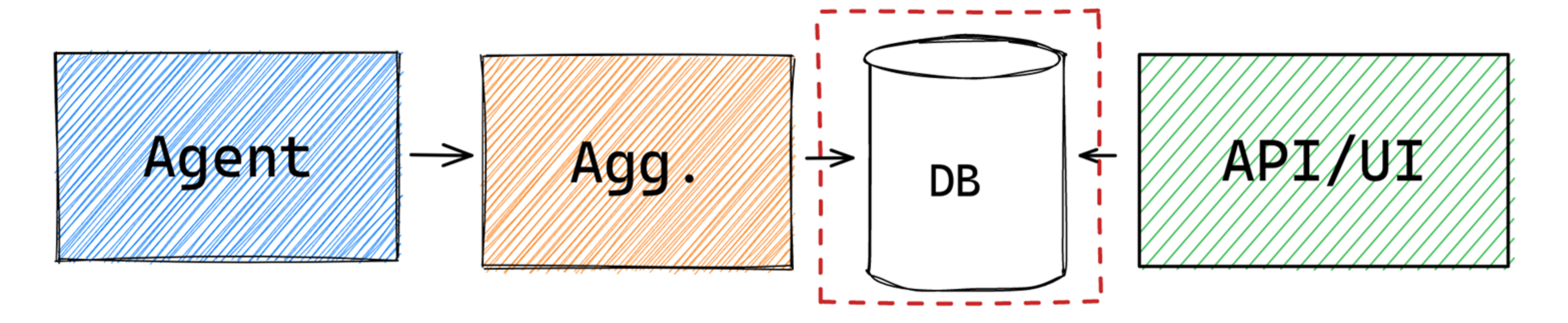
- Агрегация между процессами
- Бюджетирование + авто-семплирование
- Шардирование + Failover
- Локальное хранение

StatsHouse: агрегатор



- Агрегация между агентами
- Бюджетирование + авто-семплирование
- Приоритизация
- Backpressure

StatsHouse: база



- Репликация
- Downsampling
- Хранение + TTL
- Выполнение запросов

ClickHouse как основное хранилище

- Эксплуатация умеет в ClickHouse
- ClickHouse масштабируется
- ClickHouse не тормозит™*

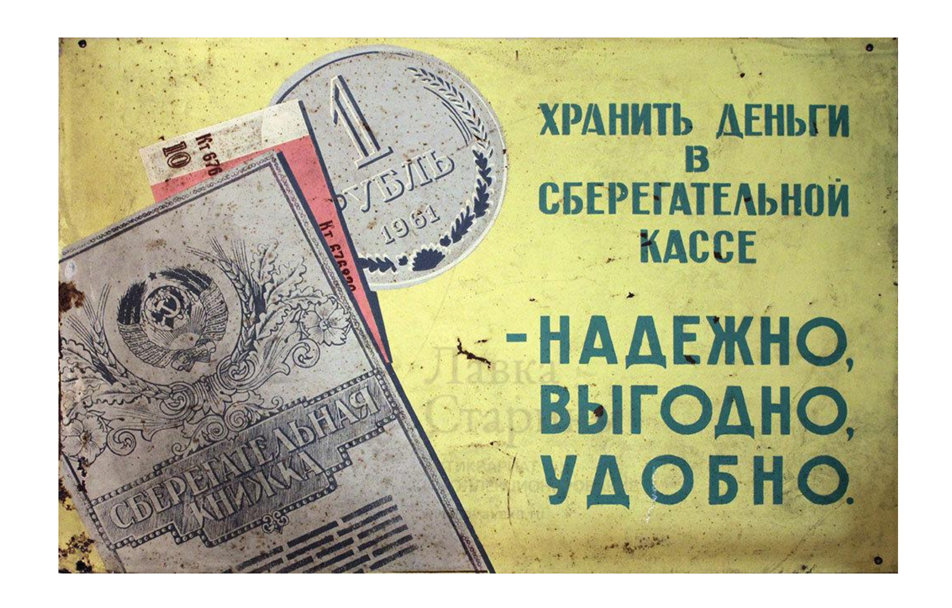
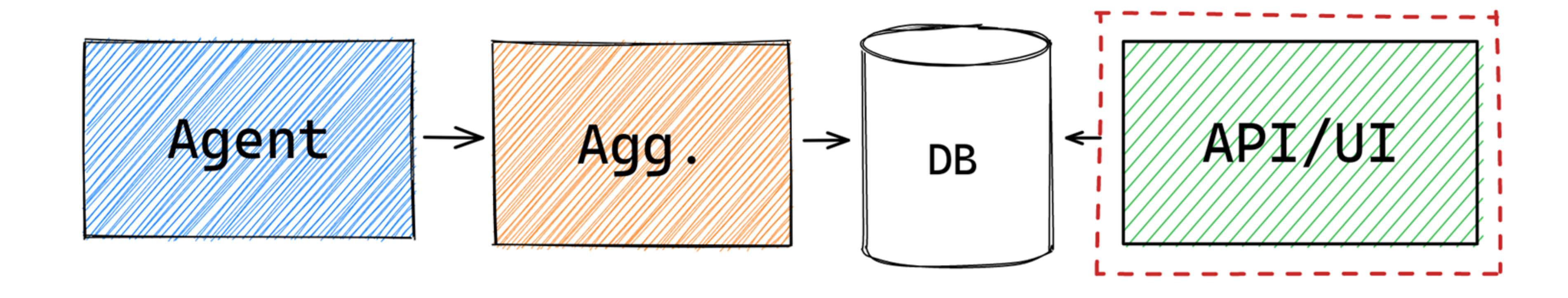


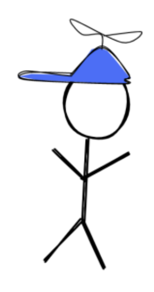
Схема хранения

time	metric	tag1	tag2	count
01:14:21	cdn_ttfb	msk	android	3015682
01:14:21	cdn_ttfb	msk	ios	936581
01:14:21	cdn_ttfb	nsk	android	1025718
01:14:21	cdn_ttfb	nsk	ios	315981
01:14:21	cdn_ttfb	spb	android	2390001

- Только 3 таблицы
  - Секундная
  - Минутная
  - Часовая
- Только 16 тегов
- Числа вместо строк

StatsHouse: API + UI





- Контроль доступа
- Ограничение нагрузки
- Кеширование

api_methods: count/sec

Вызовы API (без учета execute.* вызовов)

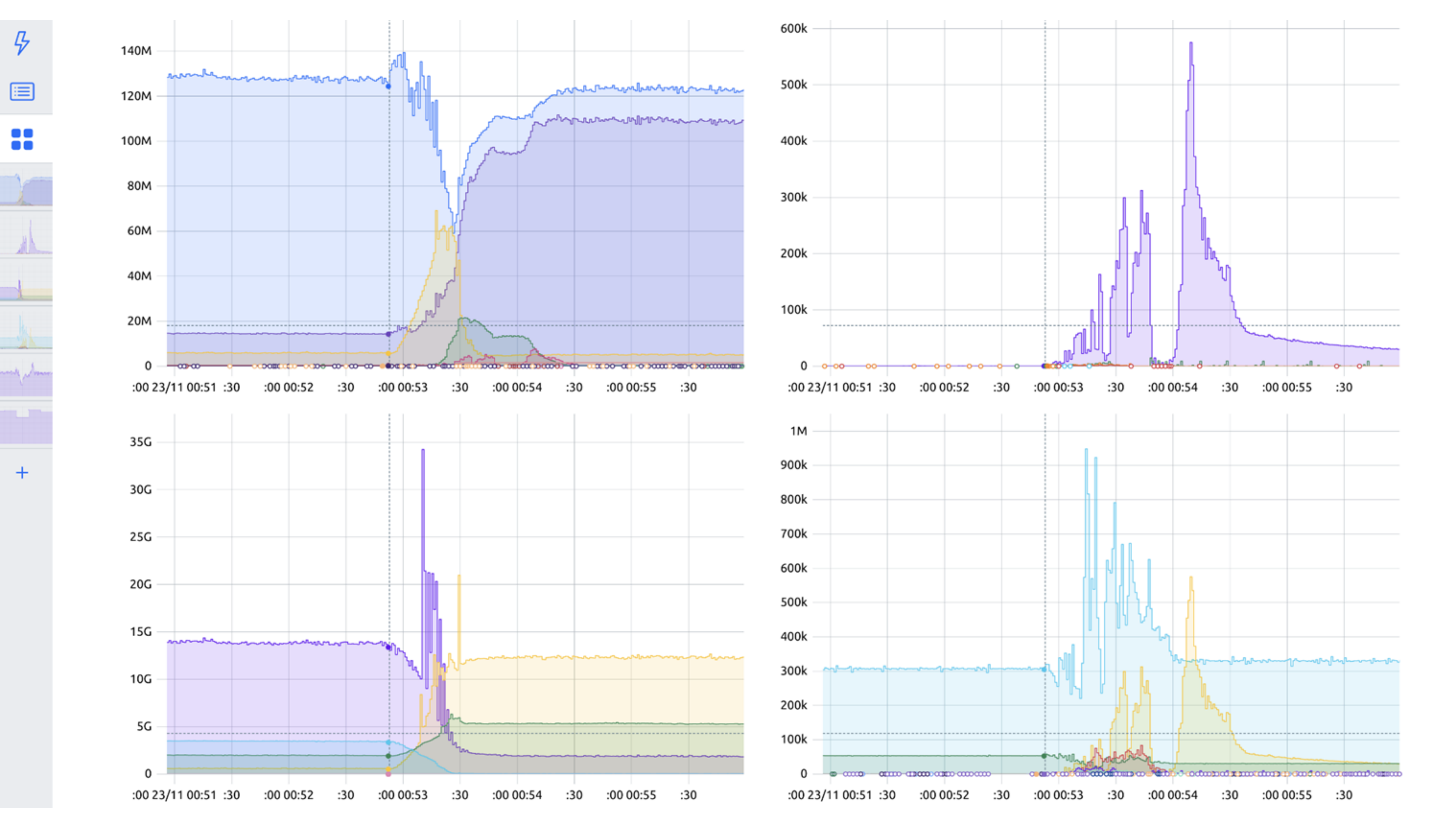
Receive status   Sampling: source / aggregator   Cardinality   Mapping status   CSV   Edit

1.2M
1.1M
1M
900k
800k
700k
600k
500k
400k
300k
200k
100k
0

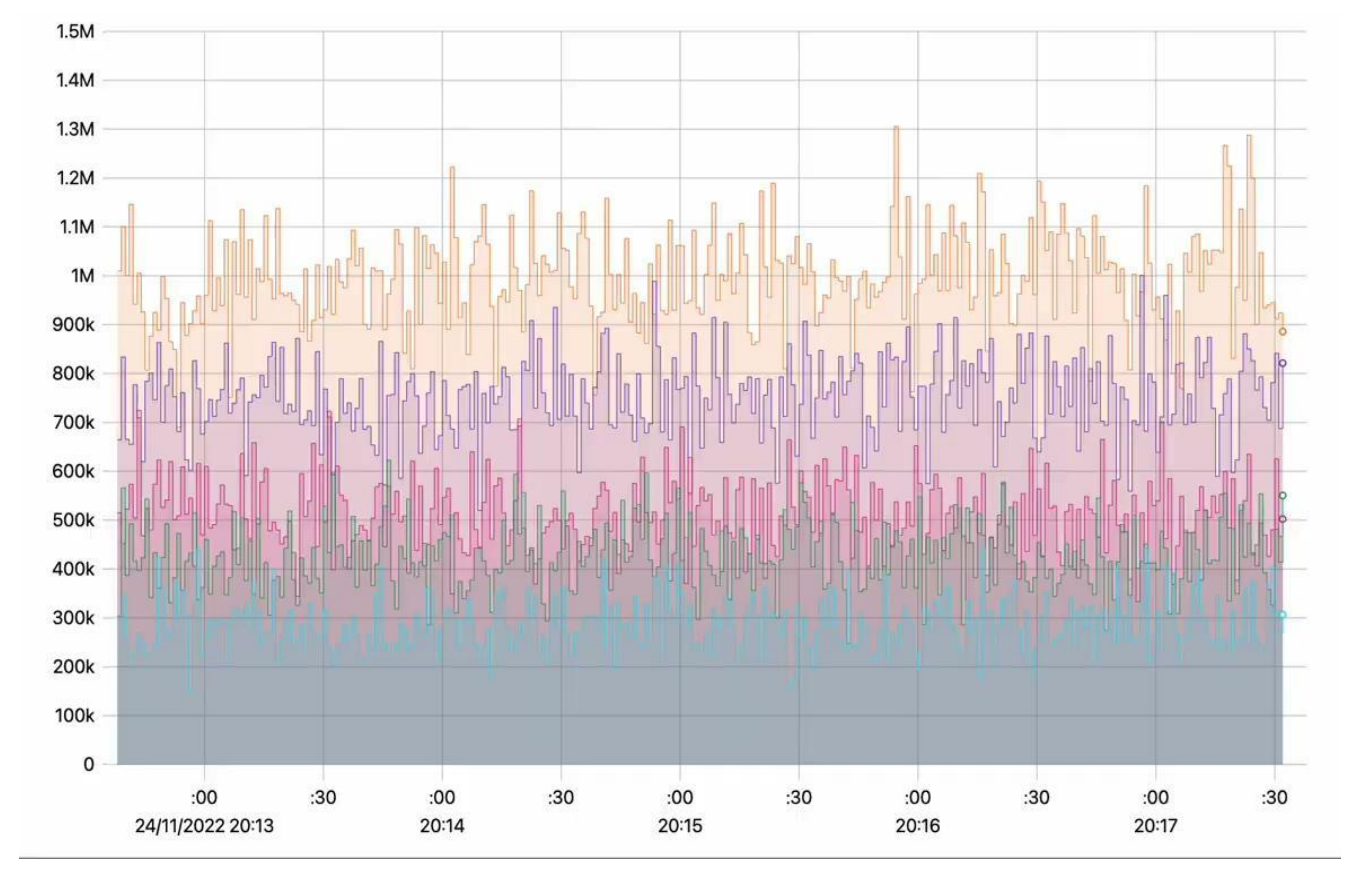
20:00 22/11/2022 | 00:00 23/11 | 04:00 | 08:00 | 12:00 | 16:00 | 20:00 | 00:00 24/11 | 04:00 | 08:00 | 12:00 | 16:00

Time —
account —
messages —
stats —

api_methods
count/sec
Last 48 hours   Today   Week
24/11/2022   16:01:43
-24h   -48h   -1w   -2w   -3w   -4w   -365d
Auto   Top 5
environment
production
группа методов
метод
версия
версия minor
платформа



140M
120M
100M
80M
60M
40M
20M
0
:00 23/11 00:51 :30 :00 00:52 :30 :00 00:53 :30 :00 00:54 :30 :00 00:55 :30
600k
500k
400k
300k
200k
100k
0
:00 23/11 00:51 :30 :00 00:52 :30 :00 00:53 :30 :00 00:54 :30 :00 00:55 :30
35G
30G
25G
20G
15G
10G
5G
0
:00 23/11 00:51 :30 :00 00:52 :30 :00 00:53 :30 :00 00:54 :30 :00 00:55 :30
1M
900k
800k
700k
600k
500k
400k
300k
200k
100k
0
:00 23/11 00:51 :30 :00 00:52 :30 :00 00:53 :30 :00 00:54 :30 :00 00:55 :30


1.5M
1.4M
1.3M
1.2M
1.1M
1M
900k
800k
700k
600k
500k
400k
300k
200k
100k
0
:00
24/11/2022 20:13
:30
:00
20:14
:30
:00
20:15
:30
:00
20:16
:30
:00
20:17
:30

StatsHouse сегодня

1 секунда
разрешение

5 секунд
задержка

4 года
данных

350 000 000
метрик в секунду

15 000
серверов-источников

Анонс!



StatsHouse — Open Source

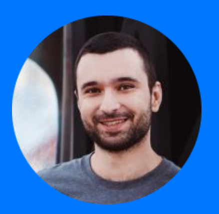
github.com/vkcom/
statshouse

MPL 2.0



StatsHouse

github.com/vkcom/statshouse









Григорий Петросян
vk.com/gp
t.me/pgregory



Запасные слайды



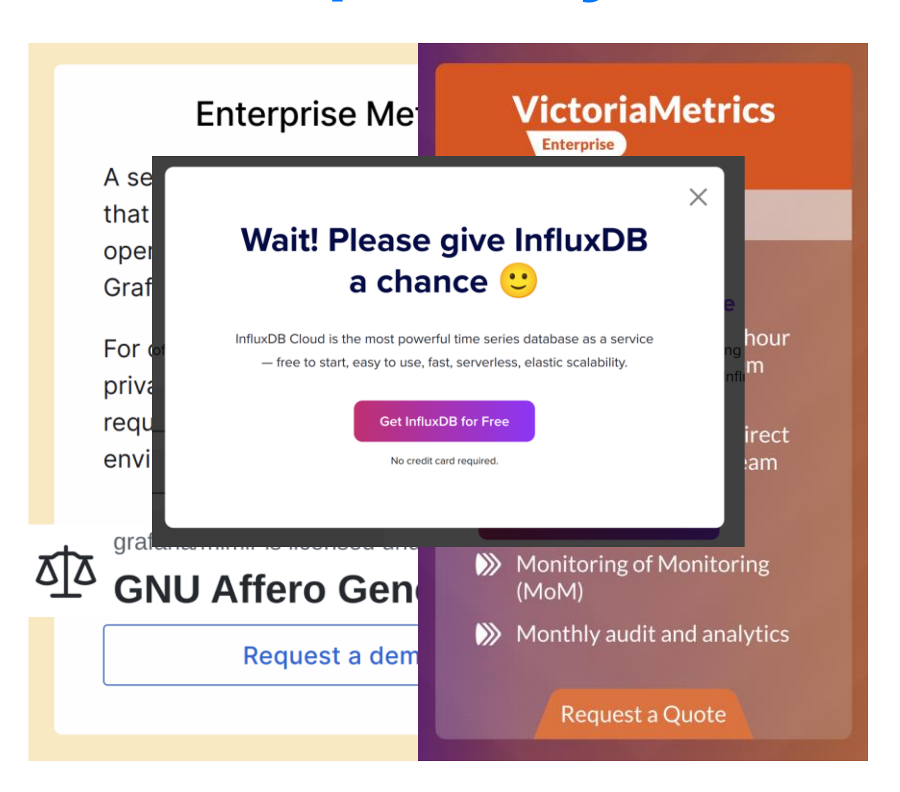
Планы

- Q1 2023
  - PromQL
  - События
- Q2 2023
  - Логи
- Q3 2023
  - Экземпляры
- Q4 2023
  - Встроенный алертинг

Attribution

- Эмоджи: OpenMoji, CC BY-SA 4.0

Мониторинг: купить или разработать



Google	Monarch
Facebook	Gorilla
Twitter	Cuckoo
Uber	M3
Netflix	Atlas
Яндекс	Solomon

StatsHouse и ClickHouse





Схема таблицы

```
CREATE TABLE statshouse_value (
    time              DateTime,
    metric            Int32,
    tag0 … tag15      Int32,
    count             AggregateFunction(sum, Float64),
    sum               AggregateFunction(sum, Float64),
    …
)
    ENGINE =  AggregatingMergeTree()
     ORDER BY (metric, time, tag0, …, tag15)
PARTITION BY toStartOfDay(time)
```

Только 16 тегов — хватит всем™

```
CREATE TABLE statshouse_value (
    time            DateTime,
    metric          Int32,
    tag0 … tag15    Int32,
    count           AggregateFunction(sum, Float64),
    sum             AggregateFunction(sum, Float64),
    …
)
    ENGINE =  AggregatingMergeTree()
     ORDER BY (metric, time, tag0, …, tag15)
PARTITION BY toStartOfDay(time)
```

Числа вместо строк — 4x быстрее

```
CREATE TABLE statshouse_value (
    time                DateTime,
    metric              Int32,
    tag0 … tag15        Int32,
    count               AggregateFunction(sum, Float64),
    sum                 AggregateFunction(sum, Float64),
    …
)
    ENGINE =  AggregatingMergeTree()
     ORDER BY (metric, time, tag0, …, tag15)
PARTITION BY toStartOfDay(time)
```

Только 3 таблицы — downsampling + TTL

```
CREATE TABLE statshouse_value_{1s, 1m, 1h} (
    time              DateTime,
    metric            Int32,
    tag0 … tag15      Int32,
    count             AggregateFunction(sum, Float64),
    sum               AggregateFunction(sum, Float64),
    …
)
    ENGINE =  AggregatingMergeTree()
     ORDER BY (metric, time, tag0, …, tag15)
PARTITION BY toStartOfDay(time)
```

Только 1 скан — оптимально

```
CREATE TABLE statshouse_value_1m (
   time               DateTime,
   metric             Int32,
   tag0 … tag15       Int32,
   count              AggregateFunction(sum, Float64),
   sum                AggregateFunction(sum, Float64),
   …
)
   ENGINE =  AggregatingMergeTree()
   ORDER BY (metric, time, tag0, …, tag15)
PARTITION BY toStartOfDay(time)
```

v1.0